alysis

Release 0.4.0

Bogdan Opanchuk

Mar 15, 2024

CONTENTS:

CHAPTER

ONE

PUBLIC API

1.1 Typed

class alysis.**Node**(*, *root_balance_wei: int*, *chain_id: int = 107118521969011*, *net_version: int = 1*, *auto_mine_transactions: bool = True*)

An Ethereum node maintaining its own local chain.

If `auto_mine_transactions` is `True`, a new block is mined after every successful transaction.

delete_filter(*filter_id: int*) → None

Deletes the filter with the given identifier.

disable_auto_mine_transactions() → None

Turns automining off.

enable_auto_mine_transactions() → None

Turns automining on and mines a new block.

eth_block_number() → int

Returns the number of most recent block.

eth_call(*params:* EthCallParams, *block: int* | BlockLabel) → bytes

Executes a new message call immediately without creating a transaction on the blockchain.

If the transaction is invalid, raises *ValidationError*. If the transaction is sent to the EVM but is reverted during execution, raises *TransactionReverted*. If there were other problems with the transaction, raises *TransactionFailed*.

eth_chain_id() → int

Returns the chain ID used for signing replay-protected transactions.

eth_estimate_gas(*params:* EstimateGasParams, *block: int* | BlockLabel) → int

Generates and returns an estimate of how much gas is necessary to allow the transaction to complete. The transaction will not be added to the blockchain.

If the transaction is invalid, raises *ValidationError*. If the transaction is sent to the EVM but is reverted during execution, raises *TransactionReverted*. If there were other problems with the transaction, raises *TransactionFailed*.

eth_gas_price() → int

Returns an estimate of the current price per gas in wei.

eth_get_balance(*address:* Address, *block: int* | BlockLabel) → int

Returns the balance (in wei) of the account of given address.

eth_get_block_by_hash(*block_hash:* Hash32, *, *with_transactions: bool*) → *BlockInfo*

Returns information about a block by hash.

Raises *BlockNotFound* if the requested block does not exist.

eth_get_block_by_number(*block: int* | BlockLabel, *, *with_transactions: bool*) → *BlockInfo*

Returns information about a block by block number.

Raises *BlockNotFound* if the requested block does not exist.

eth_get_code(*address:* Address, *block: int* | BlockLabel) → bytes

Returns code of the contract at a given address.

eth_get_filter_changes(*filter_id: int*) → list[*LogEntry*] | list[*Hash32*]

Polling method for a filter, which returns an array of logs which occurred since last poll.

Note: This method will not return the events that happened before the filter creation, even if they satisfy the filter predicate. Call *eth_get_filter_logs()* to get those.

---

eth_get_filter_logs(*filter_id: int*) → list[*LogEntry*]

Returns an array of all logs matching filter with given id.

eth_get_logs(*params:* FilterParams | FilterParamsEIP234) → list[*LogEntry*]

Returns an array of all logs matching a given filter object.

eth_get_storage_at(*address:* Address, *slot: int*, *block: int* | BlockLabel) → bytes

Returns the value from a storage position at a given address.

eth_get_transaction_by_hash(*transaction_hash:* Hash32) → *TransactionInfo*

Returns the information about a transaction requested by transaction hash.

Raises *TransactionNotFound* if the transaction with this hash has not been included in a block yet.

eth_get_transaction_count(*address:* Address, *block: int* | BlockLabel) → int

Returns the number of transactions sent from an address.

eth_get_transaction_receipt(*transaction_hash:* Hash32) → *TransactionReceipt*

Returns the receipt of a transaction by transaction hash.

Raises *TransactionNotFound* if the transaction with this hash has not been included in a block yet.

eth_new_block_filter() → int

Creates a filter in the node, to notify when a new block arrives. Returns the identifier of the created filter.

eth_new_filter(*params:* FilterParams) → int

Creates a filter object, based on filter options, to notify when the state changes (logs). Returns the identifier of the created filter.

eth_new_pending_transaction_filter() → int

Creates a filter in the node, to notify when new pending transactions arrive. Returns the identifier of the created filter.

eth_send_raw_transaction(*raw_transaction: bytes*) → *Hash32*

Attempts to add a signed RLP-encoded transaction to the current block. Returns the transaction hash on success.

If the transaction is invalid, raises *ValidationError*. If the transaction is sent to the EVM but is reverted during execution, raises *TransactionReverted*. If there were other problems with the transaction, raises *TransactionFailed*.

mine_block(*timestamp: None | int = None*) → None

Mines a new block containing all the pending transactions.

If `timestamp` is not `None`, sets the new block's timestamp to the given value.

net_version() → int

Returns the current network id.

root_private_key: bytes

The private key of the funded address created with the chain.

1.2 RPC

class alysis.**RPCNode**(*node:* Node)

A wrapper for *Node* exposing an RPC-like interface, taking and returning JSON-compatible data structures.

rpc(*method_name: str, *params: None | bool | int | float | str | Sequence[JSON] | Mapping[str, JSON]*) → None | bool | int | float | str | Sequence[JSON] | Mapping[str, JSON]

Makes an RPC request to the chain and returns the result on success, or raises *RPCError* on failure.

class alysis.**RPCError**

An exception raised in case of a known error, that is something that would be returned as `"error": {"code": ..., "message": ..., "data": ...}` sub-dictionary in an RPC response.

code: int

The error type.

data: None | bytes

The associated hex-encoded data (if any).

message: str

The associated message.

1.3 Exceptions

class alysis.**ValidationError**

Invalid values of some of the arguments.

class alysis.**BlockNotFound**

Requested block cannot be found.

class alysis.**TransactionNotFound**

Requested transaction cannot be found.

class alysis.**FilterNotFound**

Requested filter cannot be found.

class alysis.**TransactionFailed**

Transaction could not be executed.

class alysis.**TransactionReverted**

Transaction was partially executed, but had to be reverted.

1.4 Schema

alysis.schema.**JSON**

Values serializable to JSON.

class alysis.schema.**Address**

Ethereum address (20 bytes).

alias of bytes

class alysis.schema.**Hash32**

A keccak hash (32 bytes).

alias of bytes

class alysis.schema.**BlockInfo**(*number: int*, *hash: None | Hash32*, *parent_hash: Hash32*, *nonce: None | BlockNonce*, *sha3_uncles: Hash32*, *logs_bloom: None | LogsBloom*, *transactions_root: Hash32*, *state_root: Hash32*, *receipts_root: Hash32*, *miner: None | Address*, *difficulty: int*, *total_difficulty: None | int*, *extra_data: bytes*, *size: int*, *gas_limit: int*, *gas_used: int*, *base_fee_per_gas: int*, *timestamp: int*, *transactions: list[TransactionInfo] | list[Hash32]*, *uncles: list[Hash32]*)

Block info.

class alysis.schema.**BlockNonce**

Block nonce (8 bytes).

alias of bytes

enum alysis.schema.**BlockLabel**(*value*)

Block label.

Valid values are as follows:

LATEST = <BlockLabel.LATEST: 'latest'>

PENDING = <BlockLabel.PENDING: 'pending'>

SAFE = <BlockLabel.SAFE: 'safe'>

FINALIZED = <BlockLabel.FINALIZED: 'finalized'>

EARLIEST = <BlockLabel.EARLIEST: 'earliest'>

class alysis.schema.**EthCallParams**(*to: Address*, *from_: None | Address = None*, *gas: None | int = None*, *gas_price: int = 0*, *value: int = 0*, *data: None | bytes = None*)

Transaction fields for `eth_call`.

class alysis.schema.**EstimateGasParams**(*from_: Address*, *to: None | Address = None*, *gas: None | int = None*, *gas_price: int = 0*, *nonce: None | int = None*, *value: int = 0*, *data: None | bytes = None*)

Transaction fields for `eth_estimateGas`.

class alysis.schema.**FilterParams**(*from_block: None | int | BlockLabel = None*, *to_block: None | int | BlockLabel = None*, *address: None | Address | list[Address] = None*, *topics: None | list[None | LogTopic | list[LogTopic]] = None*)

Filter parameters for `eth_getLogs` or `eth_newFilter`.

class alysis.schema.**FilterParamsEIP234**(*block_hash:* Hash32, *address:* *None* | Address | *list[*Address*]* = *None*, *topics:* *None* | *list[None* | LogTopic | *list[*LogTopic*]]* = *None*)

Alternative filter parameters for `eth_getLogs` (introduced in EIP-234).

class alysis.schema.**LogEntry**(*address:* Address, *block_hash:* Hash32, *block_number:* *int*, *data:* *bytes*, *log_index:* *int*, *removed:* *bool*, *topics:* *list[*LogTopic*]*, *transaction_index:* *int*, *transaction_hash:* Hash32)

Log entry.

class alysis.schema.**LogsBloom**

Bloom filter for logs (256 bytes).

alias of `bytes`

class alysis.schema.**LogTopic**

Encoded log topic (32 bytes).

alias of `bytes`

class alysis.schema.**TransactionInfo**(*chain_id:* *int*, *block_hash:* *None* | Hash32, *block_number:* *int*, *from_:* Address, *gas:* *int*, *gas_price:* *int*, *max_fee_per_gas:* *int*, *max_priority_fee_per_gas:* *int*, *hash:* Hash32, *input:* *bytes*, *nonce:* *int*, *to:* Address, *transaction_index:* *None* | *int*, *type:* *int*, *value:* *int*, *v:* *int*, *r:* *int*, *s:* *int*)

Transaction info.

class alysis.schema.**TransactionReceipt**(*transaction_hash:* Hash32, *transaction_index:* *int*, *block_hash:* Hash32, *block_number:* *int*, *from_:* Address, *to:* *None* | Address, *cumulative_gas_used:* *int*, *effective_gas_price:* *int*, *gas_used:* *int*, *contract_address:* *None* | Address, *logs:* *list[*LogEntry*]*, *logs_bloom:* LogsBloom, *type:* *int*, *status:* *int*)

Transaction receipt.

CHAPTER

TWO

CHANGELOG

2.1 0.4.0 (2024-03-15)

2.1.1 Changed

- `RPCError.data` is now `None | bytes` instead of `None | str`. (PR_23)
- `compages` dependency bumped to 0.3. (PR_23)

2.2 0.3.0 (2024-03-09)

2.2.1 Changed

- `Node.take_snapshot()` removed, instead `Node` objects are now deep-copyable. (PR_18)
- `RPCErrorCode.INVALID_REQEST` removed. (PR_20)
- Transaction validation errors now raise `ValidationError` instead of `TransactionFailed`. (PR_20)
- `Address` and `Hash32` from `eth-typing` are now internal and are replaced with the ones defined in the `schema` submodule. (PR_22)
- All parameters for `Node` are now keyword-only. (PR_22)

2.2.2 Added

- Support for `blockHash` parameter in `eth_getLogs`. (PR_21)
- `net_version` parameters for `Node`. (PR_22)

2.2.3 Fixed

- Process transaction validation errors and missing method errors correctly on RPC level. (PR_20)
- Correctly mismatch if there are more topics in the filter than there is in the log entry. (PR_22)
- Calculate `BlockInfo.total_difficulty` correctly. (PR_22)

2.3 0.2.0 (2024-03-05)

2.3.1 Changed

- Minimum Python version bumped to 3.10. (PR_4)

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

a

INDEX

P

R

S

T

V